Modelling Business Process Variability

Marcello La Rosa
Faculty of Information Technology
Queensland University of Technology
GPO Box 2434
Brisbane QLD 4001, Australia
Phone: +61 7 3138 9488
Fax: +61 7 3138 9390
E-mail: m.larosa@qut.edu.au

Marlon Dumas
[1] Institute of Computer Science
University of Tartu
J Liivi 2
Tartu 50409, Estonia
Phone: +37 2 737 5473
Fax: +37 2 737 5468
E-mail: marlon.dumas@ut.ee

[2] Faculty of Information Technology
Queensland University of Technology
GPO Box 2434
Brisbane QLD 4001, Australia

Arthur H.M. ter Hofstede
Faculty of Information Technology
Queensland University of Technology
GPO Box 2434
Brisbane QLD 4001, Australia
Phone: +61 7 3138 9474
Fax: +61 7 3138 9390
E-mail: a.terhofstede@qut.edu.au

Modelling Business Process Variability

ABSTRACT

A reference process model represents multiple variants of a common business process in an integrated and reusable manner. It is intended to be individualized in order to fit the requirements of a specific organization or project. This practice of individualizing reference process models provides an attractive alternative with respect to designing process models from scratch. In particular, it enables the reuse of proven practices. This chapter introduces techniques for representing variability in the context of reference process models, as well as techniques that facilitate the individualization of reference process model with respect to a given set of requirements.

Keywords: variability modeling, reference process model, process configuration

INTRODUCTION

Some business processes tend to recur in different organizations or even in different industries. For example, process analysts often use the term *order-to-cash* to refer to a business process that starts from the moment a purchase order is received by a supplier, to the moment this purchase order has been fulfilled (and the supplier has received the corresponding payment). Virtually all order-to-cash processes include activities related to invoicing, delivery and payment. However, variations can be observed across order-to-cash processes. For example, an order-to-cash process for the delivery of goods (e.g. delivery of office supplies) is different from an order-to-cash process for the delivery of services (e.g. delivery of consultancy services). In the first case, there is a physical delivery that happens at a discrete point in time and the condition of the goods can be checked upon receipt. On the other hand, the delivery of a service may occur over a long period of time (say 6 months). Over this period, several invoices may be issued for the same original purchase order. Also, checking the quality of a consultancy service is often trickier than checking the quality of a box of reams of paper. Not surprisingly, the corresponding order-to-cash process models will have many differences.

But despite such differences, companies have a lot to learn from each other when it comes to analysing and re-designing their order-to-cash processes. It would be inefficient if every time a company wants to model its order-to-cash, it did so completely from scratch, without consideration for how other companies perform their order-to-cash process. In this setting, this chapter deals with the following question: *How to model business processes that are similar to one another in many ways, yet differ in some other ways from one organization, project or industry to another?* If we can do so, it then becomes possible to capture multiple order-to-cash processes in a single model. This combined order-to-cash process model can then be used as a starting point to derive order-to-cash process models for specific companies.

This idea is captured by the concept of *reference process model* . A reference process model combines a family of similar process models together. A reference process model is designed in a generic manner and is intended to be configured to fit the requirements of specific organizations or projects. Thus, it is an alternative to designing process models from scratch.

In this chapter, we will use examples taken from the film industry, in particular from the *post-production* phase of a screen project. Figure 1 shows two process models for screen post-production: *shooting on Tape* and *shooting on Film*. The modeling language used in this figure is BPMN (cf. chapter X). These process models share some commonalities, represented by the first two activities. Whether the movie is shot on Tape or on Film, post-production always starts with the preparation of the footage for edit, followed by the Offline edit. After this activity, the two practices differ in the way the edit is completed - Online if the footage is Tape, or Negmatching if the footage is Film. Online edit is a cheap editing procedure that well combines with low-budget movies typi-

cally shot on Tape. On the other hand, if the movie is shot on Film on a high-budget production, it is preferable to carry out a Negmatching instead of an Online edit, as the former offers better quality results, although it requires higher costs. This represents a variability in post-production. Depending on the type of project, one option or the other will be used.

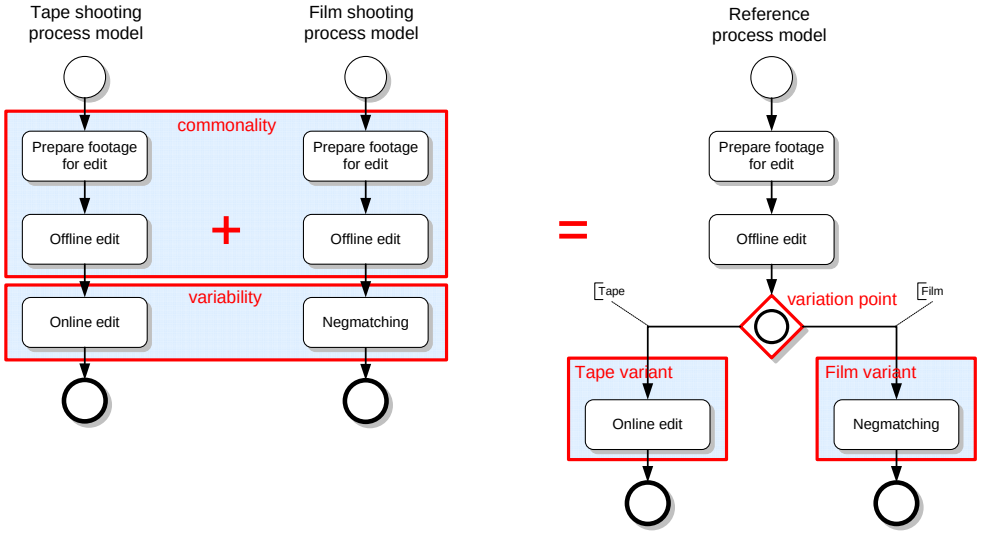

Fig. 1 A reference process model is an integrated representation of several variants of a process model.

A reference process model for screen post-production may combine both of these options, by merging the commonalities and capturing the variability by means of *variation points*. A variation point, depicted by a special OR-gateway on the right-hand side of Figure 1, is a point in the process model in which multiple variants exist and a decision needs to be taken of which variant to use. The selection of the most suitable variant is called *configuration*. Once all the variation points have been configured, the reference process model can be transformed into a derived model (e.g. by dropping the variants that are no longer needed), through a process called *individualization*.

The decision of the variants to be assigned needs to be taken before deploying and possibly executing the derived process model. This is a design-time decision, as it is not based on the availability of some data at run-time (i.e. when the process is executed), but rather on the requirements of the project or organization for which the reference process model is being configured. Hence, to leveraging reference process models in the process lifecycle (cf. chapter X), a new stage needs to be introduced, in which these decisions can be taken. This new stage, namely *configuration & individualization*, follows the design phase, where reference process models are constructed, and precedes the implementation phase, where the derived models are deployed for execution, as shown in Figure 2.

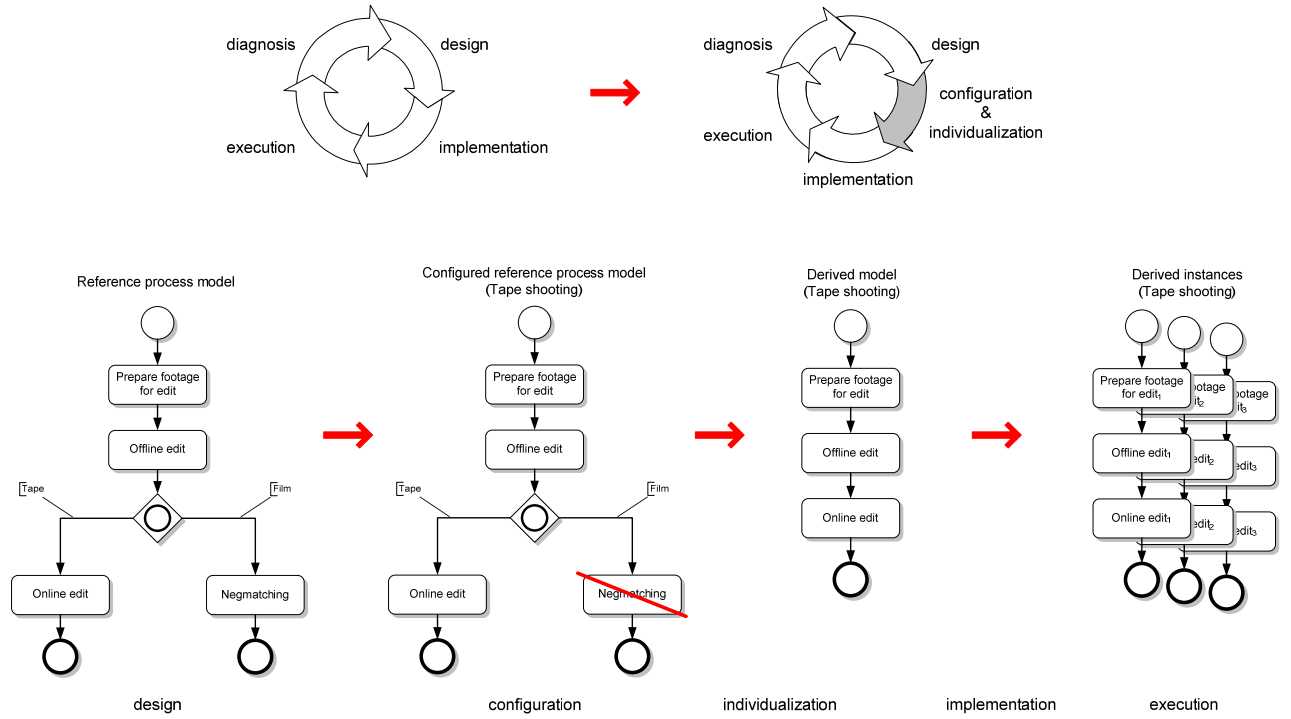

Fig. 2 Reference process models are intended to be configured and individualized to the needs of a specific setting. The *configuration & individualization* of a reference process model follows the construction phase (design time) and precedes the deployments of the derived model for execution (implementation time).

Nowadays reference process models are widespread in industry. Examples of commercial products for specific domains are ITIL[1] - for IT service management, and SCOR[2] (Stephens, 2001) - for supply chain management. The most comprehensive example is probably the SAP Reference Model[3] (Curran and Keller, 1997), incorporating a collection of common business processes which are supported by the SAP's Enterprise Resource Planning system.

Unfortunately, reference process models in commercial use tend to be captured in natural language (e.g. ITIL), or in existing general modeling languages. For example, the SAP reference process model is based on the Event-driven Process Chains (EPCs) notation (cf. chapter X). The unavailability of a dedicated reference process modeling language, with an explicit representation of variation points and variants, leads to limitations. Firstly, it is not clear which model variants exist and how they can be selected. Secondly, no decision support is provided for the actual selection of the variants, so it is difficult to estimate the impact of a configuration decision throughout the reference process model. As a result, the individualization is entirely manual and error-prone. Analysts take the reference process models merely as a source of inspiration, but ultimately, they design their own model on the basis of the reference process model, with little guidance as to which model elements need to be removed or modified to address a given requirement.

This chapter provides an overview of current research proposals aiming to address the above shortcomings. The purpose of the chapter is to show how to: (i) capture reference process models using different techniques, and (ii) capture the parameters that affect the way a reference process model will be individualized to meet specific requirements.

---

[1] http://www.itil-officialsite.com
[2] http://www.supply-chain.org
[3] http://www.sap.com/solutions/business-suite/erp

Accordingly, the first part presents several approaches for the representation of variability in business process models, based on extensions to current process modeling notations, such as EPC, YAWL (cf. chapter X) and BPMN. The second part deals with techniques to model the variability of the domain in which the reference process model has been constructed, in a way independent from the underlying process. Questionnaire models, Feature Diagrams and Adaptive Mechanisms are introduced in this part. These approaches can be used to facilitate the communication of the variability to subject-matter experts, who are usually not proficient in process modeling notations. The chapter concludes with a summary and an overview on future research trends in this field, followed by pointers to suggested readings and exercises.

LANGUAGES FOR BUSINESS PROCESS VARIABILITY MODELING

In order to capture reference process models in a reusable manner, we somehow need to represent the points in which the reference process model will differ when it is individualized. Below we present three different approaches to capture this variability. The first one is based on the concept of configurable nodes (whereby special nodes in the process can have multiple variants). In the second approach, it is rather the arcs (or edges) of the process model that are made configurable, in the sense that they can be hidden or blocked. Finally, in the third approach annotations are attached to elements of the process model to indicate in which ways they can vary during individualization.

Configurable Nodes

An approach to capture variability in process models is represented by Configurable Event-driven Process Chains (C-EPCs) (Rosemann and Aalst, 2007). C-EPCs extend EPCs by providing a means to explicitly represent variability in EPC reference process models. This is achieved by identifying a set of variation points (*configurable nodes*) in the model, to which variants (*alternatives*) can be assigned, as well as constraints to restrict the combination of allowed variants. By configuring each configurable node to exactly one alternative among the ones allowed, it is possible to derive an EPC model from the starting C-EPC.

Any function or connector can become a configurable node if it is highlighted with a thicker border in the model. Figure 3 shows a more elaborate example of the post-production reference process model in C-EPC (trivial events are omitted). Here we can identify 4 configurable functions and 5 configurable connectors. All the non-configurable nodes represent the commonalities in the reference process model. For example, function Offline edit denotes a commonality, as it is not configurable. In fact, whether the project is shot on Tape or on Film, this function will always be performed.

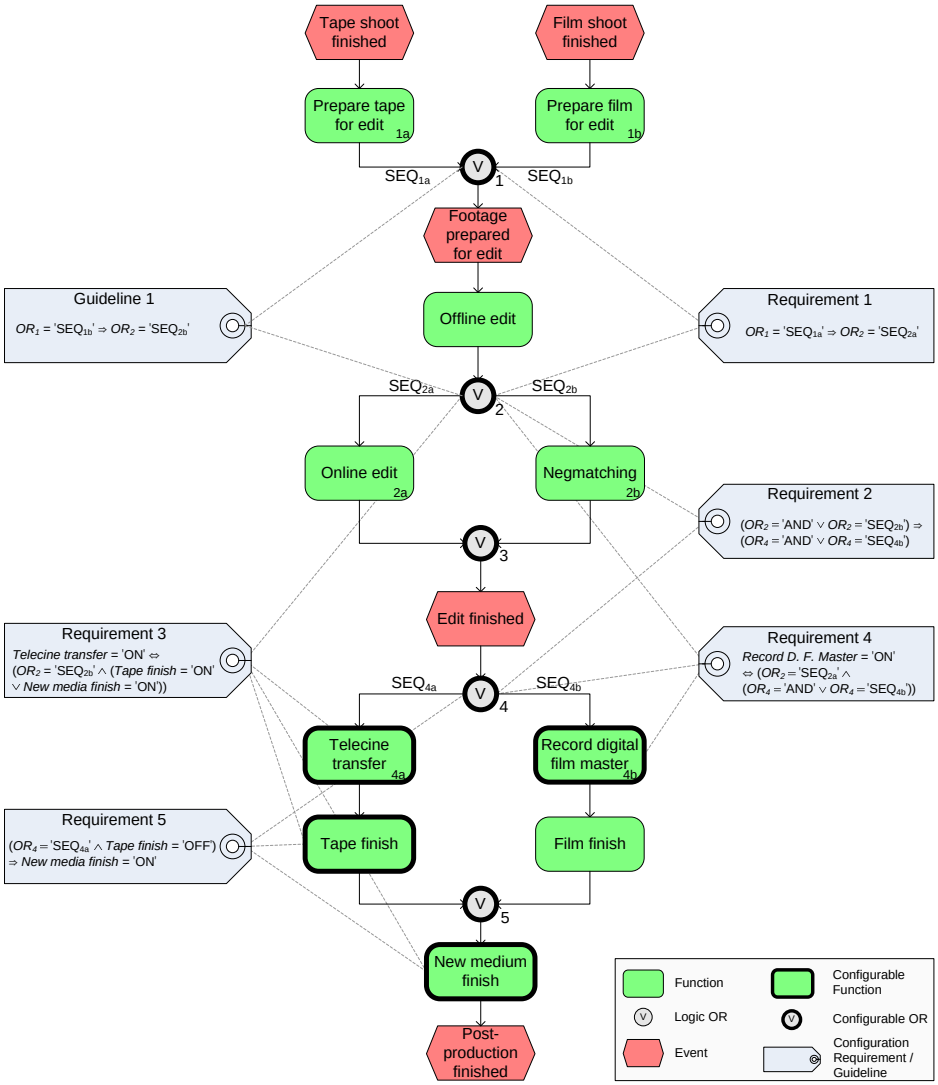

Fig. 3 The post-production reference process model in C-EPC.

Configurable functions have three alternatives: included (ON), excluded (OFF) or conditionally skipped (OPT). The first two alternatives allow one to decide a priori whether to keep the function in or permanently discard it from the process; the last option permits the deferral of this choice to run-time, where the execution of the function can be skipped on an instance-by-instance basis. For example, if we are not interested in releasing the movie on Tape, we simply need to set function Tape finish to OFF. When a function is excluded, it is removed from the process and its incoming and outgoing nodes are connected. In our case, function Telecine transfer would be directly connected to the OR-join$_5$. If a function is configured to be conditionally skipped, an XOR-split and an XOR-join are inserted in the derived model (before and after the function) to allow the user to bypass it at run-time.

Configurable connectors can only be mapped to equally or less expressive connector types. Consequently, a configurable OR can be mapped to a regular OR, XOR, AND or to one of its outgoing/incoming sequences of nodes SEQ$_n$ (where *n* is the node starting the sequence). If the connector is of type split, *n* must be one of its outgoing nodes; if the connector is of type join, *n* must be one of its incoming nodes. For example, the choice of the medium is modelled in the C-EPC of Figure 1 by configuring the OR-join$_1$. This connector can be set to its left-hand side branch - SEQ$_{1a}$ if the choice is Tape (this results in branch SEQ$_{1b}$ being removed), to its right-hand side branch - SEQ$_{1b}$ for Film (this results in branch SEQ$_{1a}$ being removed), or to an AND-join if the project sup-

ports both the media. Moreover, if the connector is configured as an OR-join, the decision of the medium is postponed to run-time, when the movie is actually shot. A configurable XOR can be set to a regular XOR or to an outgoing/incoming sequence of nodes. An AND connector can only be mapped to a regular AND, therefore not allowing any restriction. These options are summarized in Table 1.

Config. connector \ Connector type	OR	XOR	AND	SEQ_n
OR	X	X	X	X
XOR		X		X
AND			X	

Tab. 1 Configurable connectors can be configured to equally or less expressive types (Rosemann and Aalst, 2007).

Configuration requirements formalize constraints over the alternatives of configurable nodes, whilst *configuration guidelines* express advice to aid the configuration process. They are both expressed in the form of logical predicates and depicted as notes attached to the involved nodes. Only requirements are mandatory and must hold in order for a configuration to be valid. There are 5 requirements and 1 guideline in the process of Figure 1. For example, Requirement 1 (OR_1 = 'SEQ_{1a}' ⇒ OR_2 = 'SEQ_{2a}') refers to tape shooting, which implies to perform an Online edit if the medium if Tape. On the other hand, Guideline 1 (OR_1 = 'SEQ_{1b}' ⇒ OR_2 = 'SEQ_{2b}') suggests to perform a Negmatching if the shooting medium if Film, as this is the recommended procedure to get best results when the medium if Film, although an Online edit is still possible in this case.

Let us now examine the post-production reference process model in detail, as it will be used as working example throughout the chapter. Post-Production aims at the creative and technical editing a screen business project. In the first phase the footage arriving from the shooting is prepared for editing by synchronizing audio and video. The shooting medium can be Tape, Film, or both the media. Of the two, Film results in a more costly operation as special treatments are required for making it visible and permanent. Once the footage is ready, the project is edited on a low-resolution format in the Offline edit. The editing decisions are then transferred to a high-resolution format in the cut stage. The cut can be done through an Online edit and/or Negmatching, according to the shooting media (Requirement 1). This choice is modeled by configuring the OR-$split_2$ and the OR-$join_3$ to one of the two branches (SEQ_{2a}, SEQ_{2b}) or to both (AND), and is bound to the configuration of the OR-$join_1$ via Requirement 1.

After the cut stage, the project can be finished for delivery on Tape, Film, New medium (e.g. DVD, QuickTime) or any combination thereof. The overall finishing process varies on the basis of the delivery media and may involve further tasks, according to the configuration choices made before. For example since Negmatching is an expensive activity, if performed, it must lead to at least a finish of Film. This is the case of a shooting on film, and is guaranteed by Requirement 2 attached to connectors OR_2 and OR_4. Accordingly, if Negmatching is enabled by OR_2, function Film finish is always executed by forcing OR_4 to be configured to SEQ_{4b} or to an AND. On the other hand, if only Online edit is executed and the finish is on Film, function Record Digital Film Master is needed to transfer the editing results to Film. This constraint is enforced by Requirement 4. Analogously, function Telecine transfer is used only if Negmatching is performed and if a finish on Tape or New medium is expected. This is enforced by Requirement 3. Finally, Requirement 5 guarantees that at least one finish medium is selected, as function New medium finish must be set to ON, if no Film nor Tape finish is desired (i.e. if OR_4 = 'SEQ_{4a}' and Tape finish = 'OFF').

Once all the configurable nodes have been assigned an alternative that complies with the requirements, an algorithm (Rosemann and Aalst, 2007) can be used to derive an EPC from the C-EPC model. If the starting C-EPC is syntactically correct (i.e. if each node is properly connected), the algorithm ensures the preservation of the correctness in the derived model.

Let us assume we want to produce a medium budget movie on Tape, thus performing an Online edit, with a finish on Film. The corresponding configuration will be OR_1 = 'SEQ_{1a}', OR_2 = OR_3 = 'SEQ_{2a}', OR_4 = OR_5 = 'SEQ_{4b}', Telecine transfer = Tape finish = New medium finish = 'OFF', and Record digital film master = 'ON'. By applying this configuration to the C-EPC of Figure 3, we can obtain the derived model shown in Figure 4. Here the connectors have been removed as a consequence of being configured to a sequence of nodes.

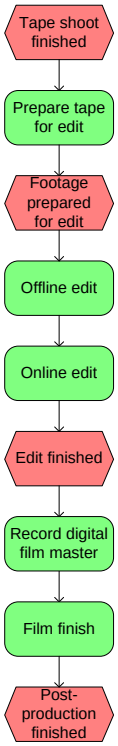

Fig. 4 The derived EPC process model from the C-EPC for a project shot on Tape, edited Online and delivered on Film.

Hiding & blocking

Another approach to capturing variability in process models in presented in (Aalst et al., 2006, Gottschalk et al., 2007a). This approach is motivated by the need for a more language-independent representation of choices in a configurable process model. In the light of this, the authors apply the operators of *hiding* and *blocking* from the concept of inheritance of workflow behaviour (Aalst and Basten, 2002) to Labelled Transition Systems (LTSs). LTSs are a formal abstraction of computing processes, therefore any process model with a formal semantics (e.g. Petri Nets or YAWL) can be mapped onto an LTS.

An LTS is a graph composed by *nodes*, representing states, and *directed edges* between nodes, representing labelled transitions, where the label can denote some event, activity or action. A traditional choice (i.e. the (X)OR in EPC or BPMN) is modelled as a node with two or more outgoing edges. Figure 6.a shows a simplified version of the post-production reference process model as an LTS, where transitions capturing the parallel execution of activities have been omitted for simplicity. For example, a choice between the edges labelled Prepare tape for edit and Prepare film for edit must be made after the first node.

Hiding and blocking can be applied to configure the edges, which are the active elements of an LTS. According to the inheritance of workflow behaviour, blocking corresponds to *encapsulation*, i.e. a blocked edge cannot be taken anymore, and the process flow will never reach a subsequent node. This implies the removal of the edge from the LTS, together with all the edges and nodes following the blocked edge, until a node with another incoming edge is reached. Hiding corresponds to *abstraction*, thus the execution of a hidden edge is skipped, but the corresponding path is still possible (the edge's label is no longer relevant). This implies the merge of the surrounding nodes.

The reference process model in Figure 5.a can be configured by selecting the desired parts through the application of hiding and blocking. Figure 5.b shows this model configured for a project shot on Tape and delivered on Film, where 3 edges have been blocked and 1 edge has been hidden. Figure 6.c shows the derived model, after the removal of the irrelevant parts and the merge of the affected nodes. The edges Offline edit, Online edit and Negmatching on the right-hand side of the LTS were not explicitly blocked, as they are removed as a consequence of blocking the edge Prepare film for edit. Similarly, the edge Tape finish has been removed after blocking Telecine transfer. On the other hand, hiding the edge New medium finish has implied the merge of its surrounding nodes.

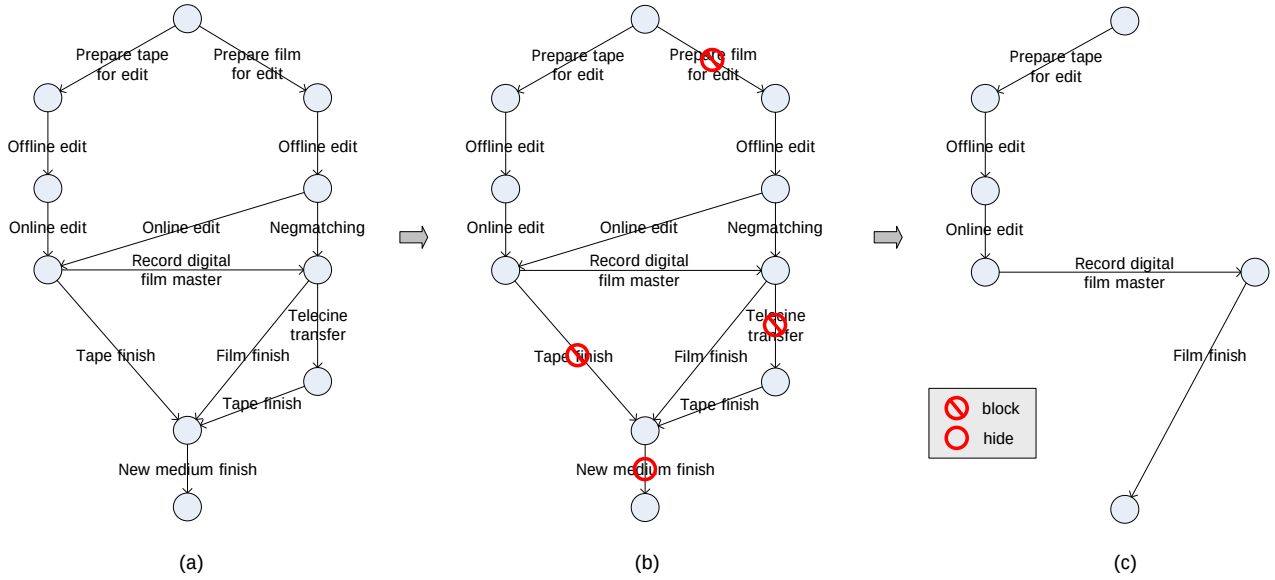

Fig. 5 Three LTSs: (a) the reference process model for post-production, (b) the configured model for a project shot on Tape, edited Online and delivered on Film, and (c) the derived model.

The option to defer decisions to run-time can be achieved using *optional blocking* and *optional hiding* (Gottschalk et al., 2007), which can be enabled at run-time.

A configurable process modelling language should allow the specification of which edges / labels can be blocked and hidden. To prove its feasibility, this approach has been applied to the executable languages of Petri Nets, YAWL, BPEL and SAP WebFlow. In this chapter we present the extension to the YAWL language, namely Configurable YAWL (C-YAWL) (Gottschalk et al., 2008).

C-YAWL extends YAWL with so-called *ports* to capture variation points (for a lexicon of the YAWL language, the reader is referred to chapter X). A task's join has an *input port* for each combination of arcs through which the task can be triggered, whilst a task's split has an *output port* for each combination of subsequent arcs that can be triggered after the task's completion. For example, let us consider the case of a join with 2 incoming arcs and a split with 2 outgoing arcs. If the join (split) is of type XOR, it will have 2 ports. This is because an XOR-join can be activated by each incoming branch, while an XOR-split will put a token only in one of its outgoing branches. If the

join (split) is of type AND, it will only have 1 port. In fact, an AND-join is activated when there is a token in both the incoming branches, while an AND-split simultaneously puts a token in all its outgoing branches. If the join is of type OR, it will only have 1 port, as the OR-join is considered as an AND-join from a configuration perspective, due to its synchronizing merge behaviour. On the other hand, an OR-split will have 3 ports - one for each combination of the outgoing arcs, as it can generate tokens for each combination of the outgoing branches.

In C-YAWL, the hiding and blocking operators are applied to input and output ports. An input port can be configured as *enabled* to allow the triggering of the task via this port, as *blocked* to prevent the triggering, or as *hidden* to skip the task's execution without blocking the subsequent process. An output port can be enabled to allow the triggering of paths leaving the port, or blocked to prevent their triggering. In C-YAWL all the ports are configurable and are enabled by default.

Figure 6.a depicts the post-production process in C-YAWL with a sample port configuration for a project shot on Tape, edited Online and finished on Film. Figure 6.b shows the YAWL model derived by applying a cleaning-up algorithm.

The first task of the model, τ_1, is used to route the process flow according to the shoot media. This task has only one incoming arc from the input condition. Therefore, its join has only one input port which always needs to be enabled (in YAWL, a task with no join/split decoration has an XOR behaviour by default). The task's OR-split has three output ports: one to trigger the path to condition 0a (leading to the preparation of the Film), one to trigger the path to condition 0b (leading to the preparation of the Tape) and one to trigger both paths. Of the three, the only port to be enabled is the one that leads to the preparation of the Tape for edit. The input port of the OR-join of the task Offline is configured as enabled as this task is always executed. Since the project is edited Online, the output port of the task Offline that triggers the condition 2b is the only one to be enabled. Similar considerations to the OR-join of Offline also hold for the OR-join of task τ_2. The project is finished on Film, so the output port of the OR-split of τ_2 that triggers 4b is the only one to be enabled. Finally, although New Medium finish is not required, the process needs to complete (a YAWL process has a unique input and output condition). Therefore, its input port is hidden.

C-YAWL allows the definition of configuration requirements to restrict the values each port can take, based on the configuration of other ports. These requirements are expressed as boolean conditions over the ports configuration, similarly to the requirements in a C-EPC model. For example, the following requirement for the post-production model binds the outgoing ports of tasks τ_1 and Offline edit, by implying to prepare the Film medium if Negmatching is to be executed: (output, (Offline edit), {2a}), enabled) $\Rightarrow$ (output, (τ_1), {0a}, enabled).

The hiding and blocking operators can also be applied to the configuration of elements specific to the YAWL language, such as cancellation regions and composite tasks. For example, it is possible to configure the cancellation region of a task by blocking the region, or to restrict the number of worklets assigned to a composite task, by blocking or hiding them. The same approach is followed for the configuration of multiple instance tasks, where the values of its parameters can be restricted, e.g., by decreasing the maximum number of allowed instances.

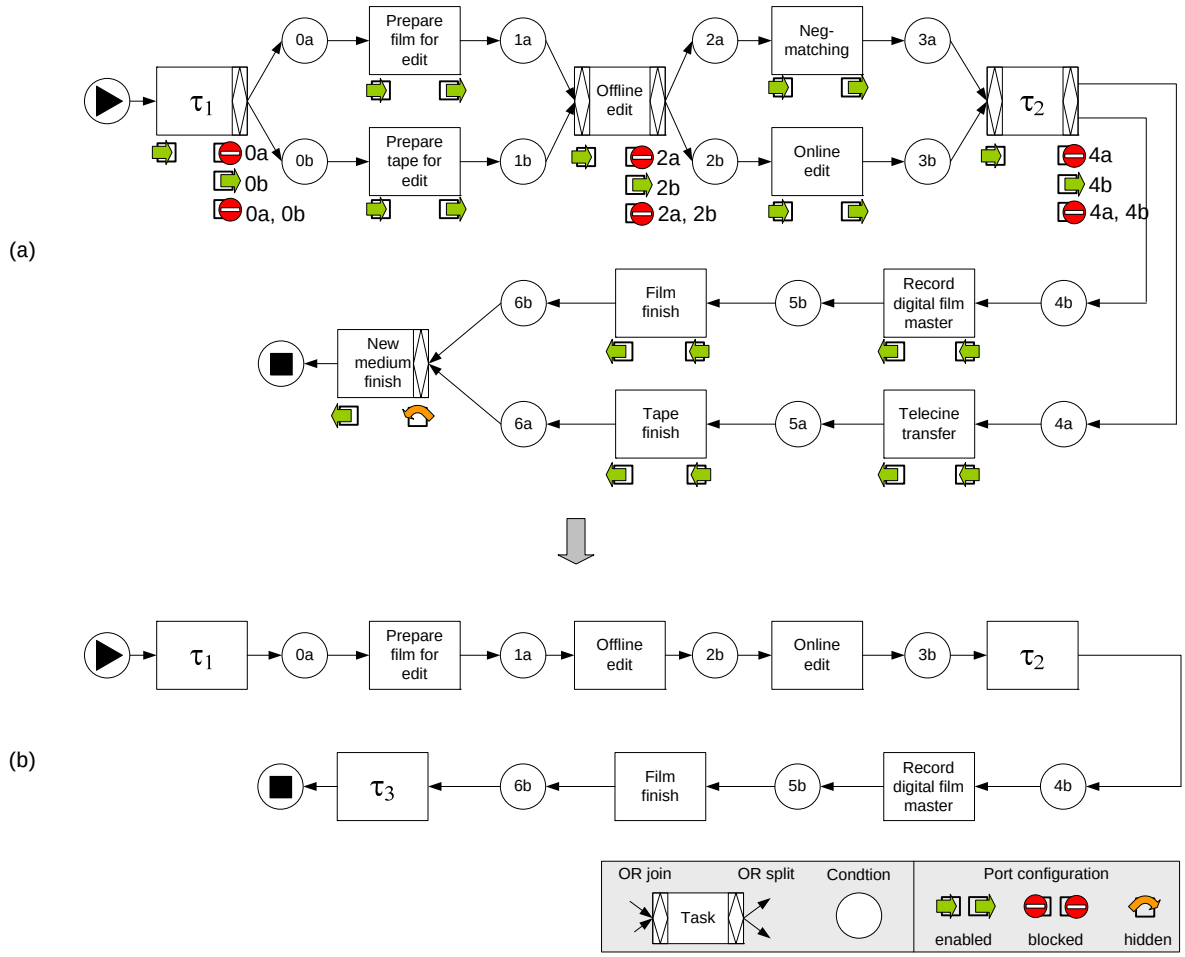

Fig. 6 (a) the post-production reference process model in C-YAWL and its port configuration for a project shot on Tape, edited Online and delivered on Film; (b) the derived YAWL model.

Annotation-based Process Variability

The idea of capturing variability in process models has also been explored in (Puhlmann et al., 2005, Schnieders and Puhlmann, 2006). The aim of this approach is not to provide a language for representing and configuring reference process models, but rather to improve the customization of process-oriented software systems, i.e. of systems that are developed from the specification of process models.

Accordingly, if the variability of a software system can be directly represented in the underlying process model, it is possible to generate code stubs for the system from the individualization of the process model itself. The purpose of this proposal is outside the topic of the chapter, so we only focus on the way the authors represent process variability.

According to this approach, a "variant-rich process model" is a process model extended with stereotype annotations to accommodate variability. Stereotypes are an extensibility mechanism borrowed from UML, that allows designers to extend the UML vocabulary with new model elements. These elements, derived from existing ones (e.g. a process activity), have specific properties that are suitable for a specific context (e.g. configuration).

A variant-rich process model can be defined in UML Activity Diagrams (ADs) (cf. chapter X) or BPMN. The places in a process model where variability can occur are marked as variation points with the stereotype «VarPoint». A variation point represents an abstract activity, such as Prepare medium for edit, which needs to be realized with a concrete variant («Variant») among a set of

possible ones. For example, Prepare medium for edit is an abstract activity which can be realized with the variant Prepare Tape for edit, or Prepare Film for edit, or both of them.

It is possible to annotate the default variant for a variation point with the stereotype «Default». Figure 8.a shows the reference process model for post-production in annotated BPMN. Here, for example, Prepare Tape for edit is the default variant for Prepare medium for edit, as this corresponds to the most common choice in this domain.

If the variants are exclusive, i.e. if only one variant can be assigned to a given variation point, the stereotype «Abstract» is used instead of «VarPoint». In Figure 7.a we assume that the variants Online edit and Negmatching are exclusive, so their variation point Picture cut has been annotated with the tag «Abstract». As a shortcut, when the variants are exclusive, the default resolution can be depicted directly on the variation point with the stereotype «Alternative».

A variation point annotated with the stereotype «Null» indicates optional behaviour. It can only be associated to one variant and its resolution is not mandatory. This is the case of the variation point Transfer tape to film which may be resolved with the variant Record digital film master, or be completely dropped from the process model. A shortcut for a «Null» variation point and its variant is achieved by depicting the variant straight on the variation point, with the stereotype «Optional». This is the case of Telecine transfer, which subsumes the variation point Transfer film to tape.

Through a configuration, each variation point is realized with one or more variants according to its type. Figure 7.b shows the derived process model for a project shot on Tape, edited Online and finished on Film.

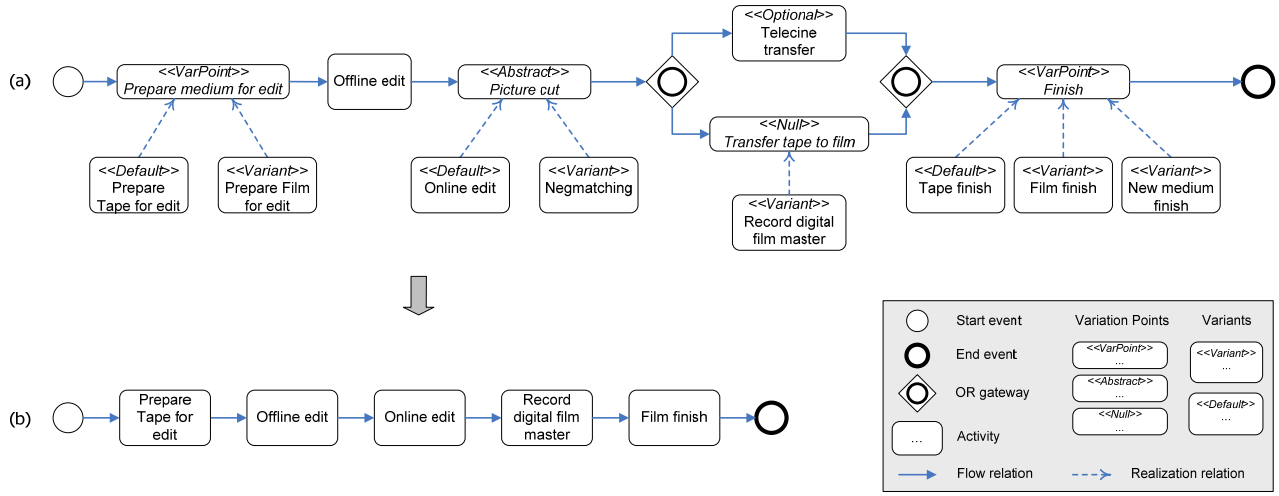

Fig. 7 (a) The post-production reference process model in annotated BPMN; (b) the derived process model.

DOMAIN-ORIENTED PROCESS CONFIGURATION

The previous section has shown different approaches to modeling variability in business processes. The purpose was to capture multiple process variants in a same artefact − the *reference process model*, which can then be configured to fit the characteristics of a specific setting. We have also seen that some approaches, like C-EPC and C-YAWL go beyond this, by capturing the dependencies among the various process variants, by means of a set of boolean expressions.

However, an aspect that is neglected by all these approaches, is the provision of support during the actual configuration of the reference process model. In other words, either there is no restriction in the number of allowed configurations (and derived models) one can obtain, or the user is left with the burden of checking the interdependencies manually. In real configuration scenarios, made up of numerous process variants (e.g. in the ITIL or SAP reference models), these interdependencies can be very complex and intricate, making the whole configuration process complex and error-prone.

Another important aspect is the relation between the reference process model and the domain in which it has been constructed. For example, in these approaches it is not clear which variation points in the reference process model are affected by a high-level decision in the context domain, e.g. shooting the project on a low budget.

Moreover, the stakeholders involved in the configuration, are required to have a thorough understanding of both the application domain and the modeling notation. While it is normal to assume that the modellers who produce the reference process model are familiar with the notation in question, it is less realistic to assume that those who provide input for configuring these models (e.g. a screen director) are sufficiently proficient with the notation.

In the light of this, this section introduces three main research proposals aiming at addressing these shortcomings. These proposals provide an independent representation of the variability in the context domain, and can be used to complement the above approaches.

Questionnaire Models

The issue of representing the variability of a given domain independently of specific notations or languages, has been dealt with in (La Rosa et al., 2007, La Rosa et al., 2008). Here the authors propose a framework based on the use of questionnaires as interfaces to configure reference process models.

In this framework, the variability of a domain is captured by a set of *domain facts*, which form the answers to a set of *questions* expressed in natural language. Each fact is a boolean variable representing a feature of the domain that can vary, i.e. that may be enabled or disabled depending on a specific application scenario. For instance, "Tape shoot" is a variant for the post-production domain, as there are projects in which this variant is enabled and others in which it is disabled (e.g. when the shooting medium is only film).

Questions group facts according to their content, so that all the facts of a same question can be set at once by answering the question. For example, the question "Which shooting media have been used?" groups the facts "Tape shoot" and "Film shoot", and allows a user to answer.

Each fact has a *default* value, which can be used to identify the most common choice for that fact. Since the majority of production projects are shot on tape, which is less expensive than film, we can assign a default value of *true* to "Tape shoot", and of *false* to "Film shoot". Moreover, a fact can be marked as *mandatory* if it needs to be explicitly set when answering the questionnaire. If a non-mandatory fact is left unset, i.e. if the corresponding question is left unanswered, its default value can be used to answer the question. In this way, each fact will always be set, either explicitly by an answer or by using its default value.

Questions and their facts are organized in a questionnaire model. Figure 8 shows one such a model for the post-production domain, where all questions and facts have been assigned a unique identifier and a description.

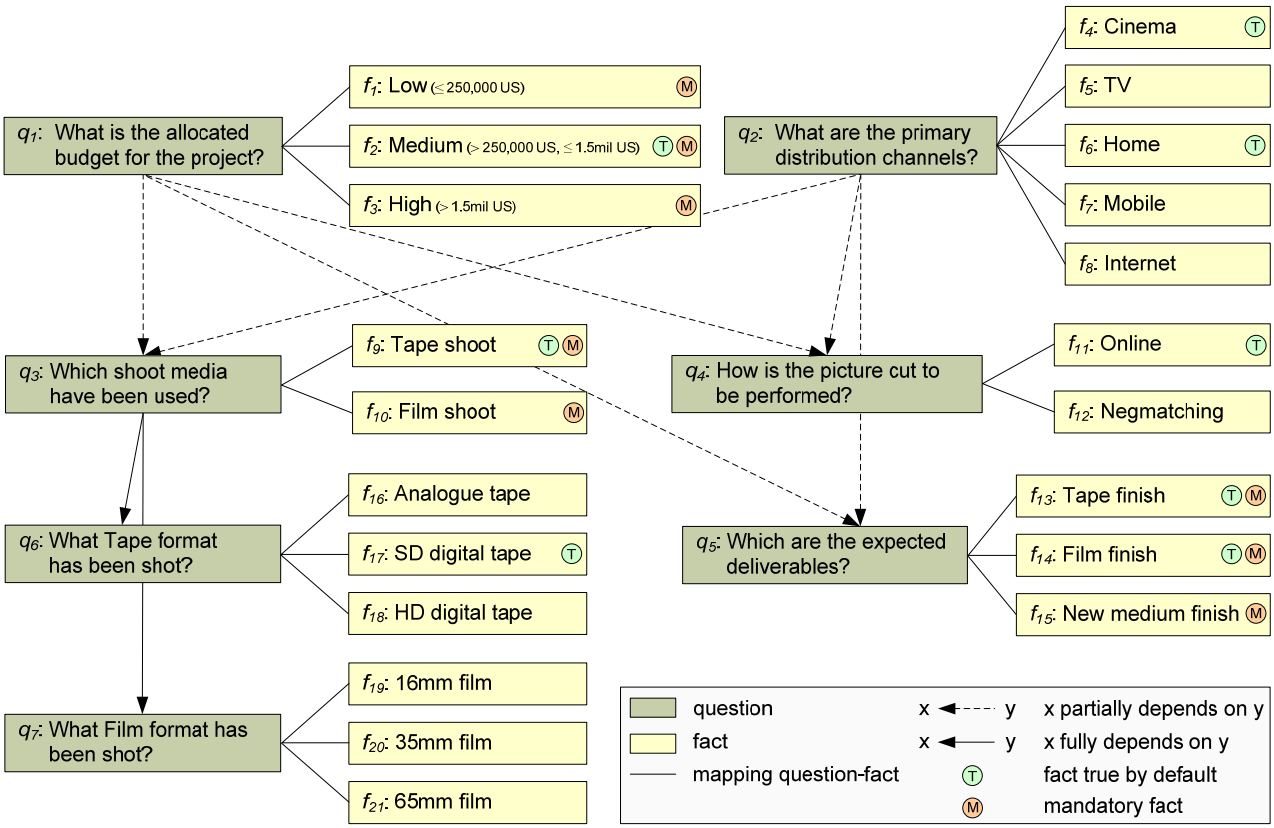

Fig. 8 A possible questionnaire model for the post-production domain.

Some questions refer to high-level decisions in post-production: question q_1 enquires the estimated budget for the project, with facts f_1 to f_3 referring to typical budget ranges for a Post-Production project. Meanwhile, question q_2 refers to the distribution channel, which can be Cinema, TV, Home, Mobile and/or Internet (each one specified by a fact). Other questions refer to the shooting media used (q_3), the type of picture cut (q_4) and the type of finish (q_5). Some other questions refer to very specific aspects of the project, such as the format of the tape (q_6) and of the film (q_7) used for the shooting.

Clearly, the facts of these two questions are related to the ones of question q_3, as it would make no sense to ask for the tape format if tape is not chosen in q_3. Similarly, there is a relation between the film formats and the choice of film in q_3. Another interplay is the one between q_3 and q_4, i.e. between the shooting medium and the picture cut. In fact, Negmatching is a costly operation and can be chosen only if the project is at least shot on film, i.e. if f_{10} is enabled in q_3. All these choices are also related to the level of budget and to the distribution channel. For low budget productions (f_1 = *true*), shooting on film (f_{10}) and finishing on film (f_{14}) are not allowed, hence their facts need to be disabled. In turn, if shooting on film is not allowed (f_{10} = *false*), Negmatching must be denied (f_{12} = *false*). Furthermore, if finishing on film is not allowed (f_{14} = *false*), distributing on Cinema must be denied (f_4 = *false*), as the latter requires a finish on film. For a medium budget, although it is allowed to finish on film, it is still not possible to shoot on film and cut with Negmatching.

These interactions among the facts of a questionnaire can be modelled with *domain constraints* in the form of boolean expressions. For example, the interactions depending on low budget can be modelled by the constraints $f_1 \Rightarrow \neg(f_{10} \vee f_{14})$, $\neg f_{10} \Rightarrow \neg f_{12}$ and $\neg f_{14} \Rightarrow \neg f_4$, while the ones on medium budget by the constraints $f_2 \Rightarrow \neg f_{10}$ and $\neg f_{10} \Rightarrow \neg f_{12}$. Since only one level of budget is allowed per project, we also need to impose the further constraint $xor(f_1, f_2, f_3)$.

Therefore, a stakeholder needs to allow for these constraints while answering the questionnaire. Therefore, a *domain configuration* is a valuation of facts as a result of completing a questionnaire, which does not violate the constraints.

In the model of Figure 8, some of the facts have been identified as mandatory to force the user to explicitly answer the questions these facts belong to. For instance, this is done for q_1, as the choice of the budget is rather important and cannot be neglected by the user. Moreover, default values have been assigned in order to reflect the typical choices made in a medium budget project, with Cinema and Home video distribution.

A questionnaire model also establishes an order relation for posing questions to the user. This is done via order dependencies. A *partial dependency* (represented by a dashed arrow) captures an optional precedence between two questions: e.g. q_3 can be posed after q_1 or q_2 have been answered. A *full dependency* (full arrow) captures a mandatory precedence: e.g. q_6 is posed after q_3 only. The dependencies can be set in a way to give priority to the most discriminating questions, i.e. q_1 and q_2, so that subsequent questions can be (partly) answered by using the constraints. If, e.g., we answer q_3 with "Film shoot" only, the question about the tape formats (q_6) becomes irrelevant, and so on. These dependencies can be arbitrary so long as cycles are avoided.

The questionnaire model, constructed by a team of modellers in collaboration with domain experts, can be linked to the variation points of a reference process model with the purpose of configuring and individualizing the latter. Users (e.g. a subject-matter expert) can answer the questionnaire by means of an interactive tool that poses the questions in an order consistent with the order dependencies, and prevents the user from entering conflicting answers to subsequent questions by dynamically checking the constraints. In this way the user is not left with the burden of manually checking the constraints among the variants of the domain. Also, questions are in natural language, fostering the user to reason directly in terms on domain concepts, rather than modeling elements. Once the questionnaire has been completed, the answers can be collected and used to automatically individualize the reference process model, as shown in Figure 9.

The idea is that each variation point and its variants in a reference process model can be associated with boolean expressions over the facts of a questionnaire model. Such expressions embody the requirements of the configurable process and the constraints of the domain. Thus, an alternative is selected whenever the corresponding boolean expression evaluates to *true*, triggering the execution of an action to configure the variation point with the selected variant, and to remove the irrelevant variants.

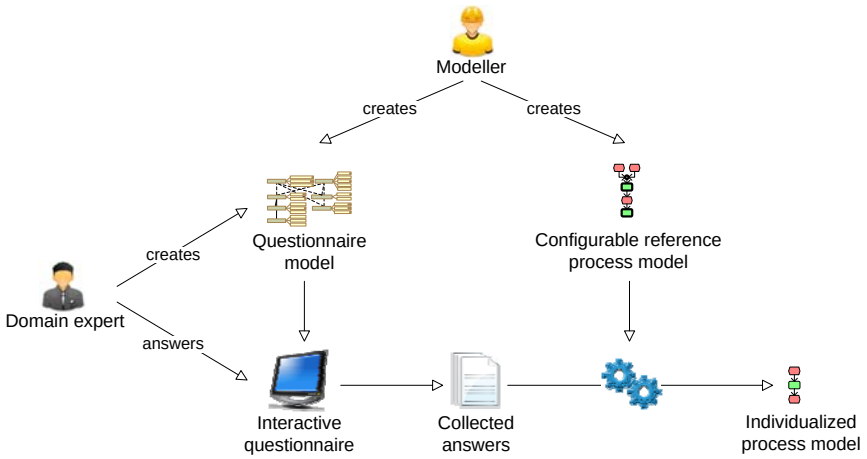

Fig. 9 The questionnaire-based framework for reference process models configuration.

The approach has been implemented in an interactive tool called *Quaestio*[4] and tested with reference process models defined in the C-EPC and C-YAWL languages.

Let us see how the link between a questionnaire model and a reference process model works, by considering the C-EPC example for post-production shown in Figure 3. We can notice that a mapping can be established between question q_4: "How is the picture cut to be performed", and the variants of the configurable node OR_2. This mapping should be defined in a way that i) when both f_{11} ("Online cut") and f_{12} ("Negmatching") are enabled, the node is configured as an AND, ii) when only f_{11} is enabled, the node is configured with its left-hand side branch, and iii) when only f_{12} is enabled, the node is configured with its right-hand side branch. Therefore, we need to link a boolean expression over the facts f_{11} and f_{12} to the variants of the OR_2, in order to capture the relation depicted in Figure 10. The mapping we are looking for is presented in Table 2.

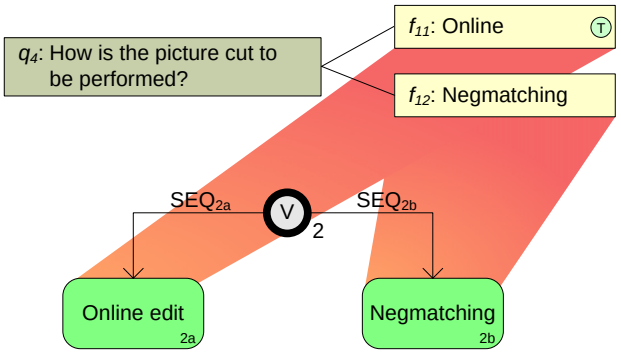

Fig. 10 The relation between the facts of question q_4 and the configurable node OR_2.

Configurable node	Alternative	Boolean expression
OR_2	AND	$f_{11} \wedge f_{12}$
	SEQ_{2a}	$f_{11} \wedge \neg f_{12}$
	SEQ_{2b}	$\neg f_{11} \wedge f_{12}$
	OR	*false*
	XOR	*false*

Tab. 2 The mapping between q_4 and OR_2.

The above is an example of direct mapping between one question and one variation point. More complex mappings can however be defined. For example, there can be questions affecting a number of variation points. In general, the most discriminating questions have a huge impact on a reference process model. This is the case of q_1. For instance, if we set the budget level to low, a number of configurable nodes in the C-EPC model would need to be configured. These are OR_1, which would be configured to SEQ_{1a} (preparation of the tape for edit), OR_2 with SEQ_{2a} (Online edit), OR_4 with SEQ_{4a} (to deny a finish of film) and Telecine transfer to OFF. Some other nodes, e.g. the configurable functions Telecine transfer and Record digital film master, are not directly affected by a specific question. Their configuration in fact depends on the answers given to questions q_4 and q_5. Record digital film master is set to ON if the cut is only done Online (f_{11} = *true* and f_{12} = *false*) and a film finish is required (f_{14} = *true*). Telecine transfer is set to ON if the cut is only done with

[4] The tool can be downloaded from http://www.processconfiguration.com

Negmatching (f_{11} = *false* and f_{12} = *true*) and the project is finished on tape (f_{13} = *true*) and/or on new medium (f_{15} = *true*).

Feature Diagrams

Another research stream has led to techniques for capturing domain variability in terms of the supported features. A number of feature modeling languages have been proposed in this field; for an overview, the reader is referred to (Schobbens et al., 2006).

These languages view feature models as tree-like structures called *feature diagrams*, with high-level features being decomposed into sub-features. A *feature* represents a domain property that is relevant to some stakeholder and is used to capture a commonality or discriminate among different domain scenarios. For example, in post-production, the feature "Edit" can be modelled with two sub-features: "Offline" and "Cut", which represent the two stages that are needed to accomplish the edit of a movie. In particular, "Offline" represents a commonality, i.e. an aspect of the domain that does not vary, while "Cut" can be further decomposed into two sub-features: "Online" and "Negmatching", representing the two possible variants for this activity.

A *feature model* consists of one or more feature diagrams and includes a set of *attributes* such as feature descriptions, constraints and mandatoriness/optionality. Constraints are arbitrary propositional logic expressions over the values of features, specified by means of a proper grammar. Constraints among the sub-features of a same feature can be graphically represented to model restrictions in the number of sub-features the feature can have. These relations can be: AND (all the sub-features must be selected), XOR (only one sub-feature can be selected) and OR (one or more can be selected). OR relationships can be further specified with an n:m cardinality, where n indicates the minimum and m indicates the maximum number of allowed sub-features. For example, the sub-features of "Cut" are bound by an OR relation (it is possible to have more that one type of cut), while the sub-features of "Budget", which are "Low", "Medium" and "High", have an XOR relation.

Figure 11 shows a possible feature diagram for the post-production domain, using the notation proposed in (Batory, 2005). There are features related to the options for budget, shooting, type of edit and transfer, finish and distribution channel.

Some features have been identified as *mandatory* (with a full circle on top of them) if they are required, while some others have been identified as *optional* (with an empty circle), if they can be excluded. The feature "Transfer" and its sub-features "Telecine" and "Digital film mastering" are optional. Their inclusion depends on the selection of the sub-features of "Edit" and "Finish", by means of proper constraints. If an optional feature always represents a variability, on the other hand, a mandatory feature does not necessarily represent a commonality. In fact, a mandatory feature can still be excluded if it has an XOR/OR relation with the sibling features. This is the case of the sub-features of "Budget", which are all mandatory (a choice on the budget is required), but only one can be included at a time, due to their XOR relation.

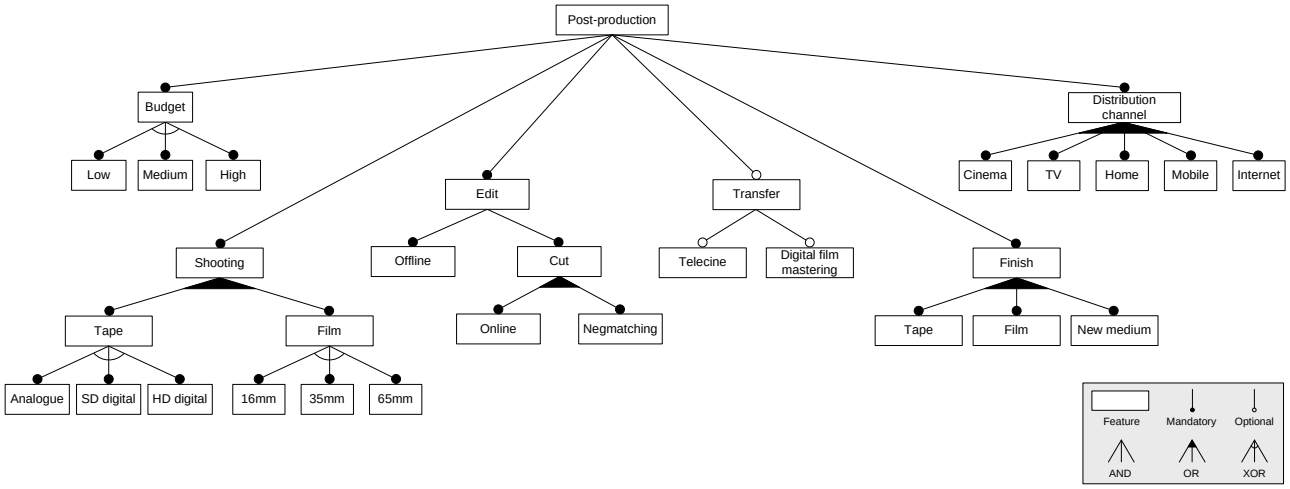

Fig. 11 A possible feature diagram for the post-production domain.

A *configuration* specifies a valid scenario in terms of features selected/deselected, viz. a scenario that complies with the constraints. Although the initial aim of feature-based approaches was to facilitate the configuration of software product families, a feature diagram can also be used for the configuration of reference process models. For example, in (Puhlmann et al., 2005) the authors link a feature diagram to a variant-rich process model in UML ADs or BPMN, by tagging each process variant with the name of a feature. In this way, the realization of variation points with variants is done via the evaluation of a feature configuration. Figure 12 shows an example of "tagged" variants for the BPMN model in Figure 7, in relation to the features of Figure 11.

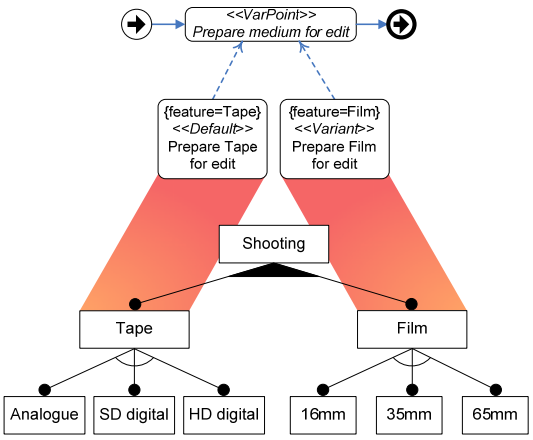

Fig. 12 The relation between the variation point *Prepare medium for edit* and the sub-features of "Shooting".

Adaptive Mechanisms

The separation of process configuration from the context domain has also been investigated in (Becker et al., 2004, Becker et al., 2006). This approach is based upon the principle of model projection. Since the reference process model contains information for multiple application scenarios, it is possible to create a projection for a specific scenario, by fading out those process branches that are not relevant to the scenario in question.

Business characteristics can be used to determine the available application scenarios. For example, in the case of post-production, we can identify the business characteristic 'Budget Level' (BL) yielding the following scenarios: 'Low budget' (L), 'Medium budget' (M) or 'High budget' (H). Another example is the 'Shooting type'), which can be 'Tape shooting' or 'Film shooting'.

The business characteristics are linked to the elements of a reference process model by means of adaptation parameters, defined in the form of simple attributes or logical terms. The language chosen by the approach to capture reference process models is (plain) EPC. Figure 12.a shows the post-production example, where each function and event is associated to a logical term referring to the project's budget. For example, the event Film shoot finished and the function Prepare film for edit are linked to the term NOT BL (L), meaning that these elements are not suitable for a low budget project, while the function Online edit has the term BL (L | M | H), meaning that it is suitable to any type of budget (where | stands for the logical OR).

The projection of the reference process model to a specific scenario is done by removing those elements whose parameters evaluate to false. Figure 12.b shows the projection of the post-production model for a low budget project.

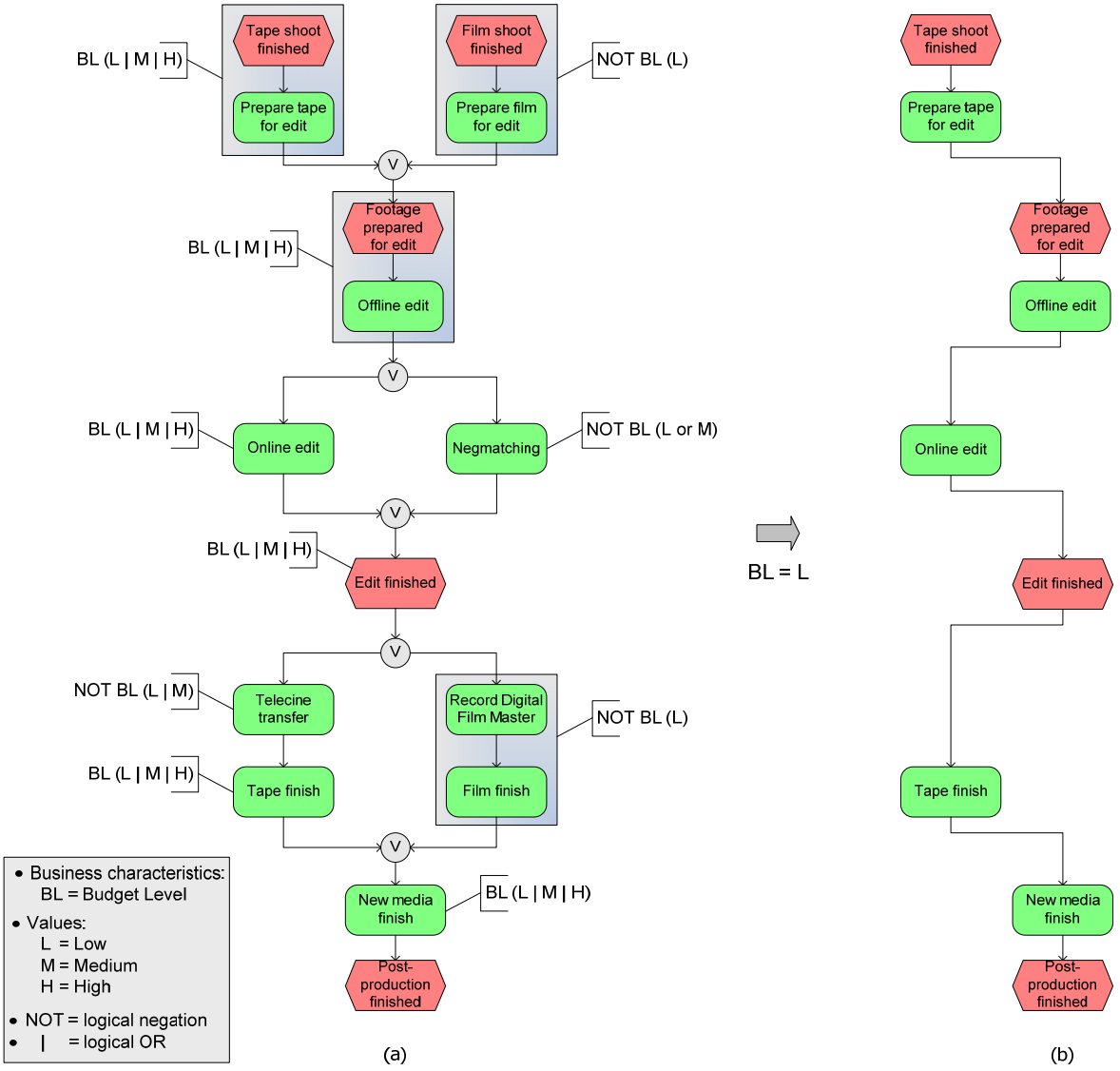

Fig. 12 (a) The post-production reference process model in EPC, with logical terms for the budget levels; (b) the model projection for a low budget project.

SUMMARY AND OUTLOOK

Reference process models constitute a promising approach to achieve reuse during process modelling. The idea is that instead of designing process models from scratch, we can take a reference process model and manipulate it to meet specific requirements. However, in current mainstream practice, reference process models tend to focus on the commonalities within a family of process models (e.g. the activities that are common to multiple order-to-cash process models). Also, existing reference process models need to be individualized manually, and they provide little guidance to modellers regarding which model elements need to be removed or modified to address a given requirement.

This chapter provided an overview of recent research proposals that aim to address these limitations in existing reference process models. On the one hand, the chapter described three techniques for capturing variability in a reference process model, so that it becomes possible to represent (in an integrated manner) which elements are common to all individualizations of a reference process model, and which elements differ (and how they differ). This basically tells us "how the reference process model varies", but not "why it varies". The second part of the chapter described techniques to capture the domain parameters that affect the variability (and therefore the configuration) of a reference process model. This is a key component of a reference process model since it provides a basis for a user to decide how the reference process model should be individualized to meet specific requirements. Three methods for capturing this domain variability were presented: one based on feature models, the other based on questionnaires, and the last based on adaptive mechanisms.

One key question that has perhaps not been sufficiently well addressed in the literature on reference process models is *how do we come up with the reference process model in the first place?* One possible starting point is to collect a number of related process models from different (preferably successful) process design projects, and to merge them together. But how can this merger be facilitated is an open question. Since process models are usually represented as graphs, techniques from the field of graph matching could come to the rescue (Bunke, 2000). For example, there exist graph matching algorithms that take as input collections of graphs and compute their similarities and differences. These techniques could be employed to identify elements that are common to all models in a collection of similar models (i.e. a common denominator) and to identify variations with respect to such a common denominator. These variations can then be captured as configurable nodes. The output of these techniques could be taken as a starting point in the design a reference process model, but of course, further information would need to be added, especially information related to the domain parameters and how these domain parameters relate to the configurable nodes in the reference process model.

Another direction for automating the construction of reference process model is by using process mining techniques. The idea of process mining is to take event logs related to a business process (e.g. all events related to an order-to-cash process) and to derive a process model that matches the event log in question. In (Jansen-Vullers et al., 2006) the authors discuss extensions to existing process mining techniques that allow one to derive a C-EPC from a regular EPC and one or several logs (extracted for example from an SAP system). The authors also show how to automate the individualization of C-EPCs using process mining techniques. Specifically, given a C-EPC and a log, their technique can derive a regular EPC corresponding to an individualization of the C-EPC. Further research is required to refine these techniques and to validate their applicability in practice.

EXERCISES

1. Describe the key benefits of configurable reference process models.

2. In a C-EPC, what are the implications of turning a "configurable OR-split" into an XOR-split during configuration? Are the resulting changes local, or do they affect other parts of the model? What about turning a configurable OR-split into an AND-split?

3. Download the Quaestio configuration tool from www.processconfiguration.com. The Quaestio tool distribution includes a screen post-production C-EPC similar to the one presented in this chapter. This C-EPC is captured as an EPML (EPC Markup Language) file, and can be viewed using a toolset known as EPCTools (http://wwwcs.uni-paderborn.de/cs/kindler/research/EPCTools).[5] The Quaestio tool distribution also includes a questionnaire model corresponding to the post-production C-EPC (see file with extension `.cml`). Load this questionnaire model into the Quaestio tool and follow the questions to individualize the configurable process model according to the following parameters:

   – High budget

   – Shooting on film

   – Distribution on cinema and home.

   After responding to all relevant questions you will obtain a *configuration*, i.e. an assignment of values to each domain fact. This configuration can then be saved and *applied* to the post-production C-EPC using the corresponding menu options in the Quaestio tool. The result will be an individualized EPC (a new EPML file) that can be displayed using EPCTools. What differences do you observe between the original C-EPC and the individualized EPC? Which tasks or branches have been removed? Compare these changes with the specification of the mapping between the questionnaire model and the C-EPC, which can be found in the file with extension `.cmap`.

4. Consider the process of travel applications in a university consisting of Faculties X and Y. In Faculty X staff members obtain a quote for the trip and provide justification in the form of supporting documentation (e.g. a letter of invitation or notification of acceptance of a conference article). They then approach their group leader for approval followed by the Dean of the Faculty. A similar process occurs in Faculty Y except that the Dean can approve the travel application before the group leader. In Faculty X if the duration of the trip exceeds 10 working days, a Faculty-based committee also needs to approve the trip and requires further information, e.g. the applicant needs to provide details of other trips they made in the last three years and be more detailed in terms of benefits that the trip may bring to the Faculty. This again is similar in Faculty Y except that this process only applies in case the trip exceeds 15 working days and no details are requested from the applicant of previous trips as the system of the Faculty provides this information automatically. Can you model both processes in a single configurable process model (e.g. as a C-EPC or C-YAWL model)? What are the features or domain facts that affect the configuration of the resulting configurable process model? What would be the benefits of represent-

[5] The EPCTools is not able to show the difference between configurable and non-configurable nodes: all nodes have the same appearance, whether they are configurable or not. To see which nodes are configurable and which ones are not, you can inspect the EPML file using an XML viewer or editor.

ing the processes at both Faculties together in a single configurable reference process model as opposed to representing them as two completely separate process models?

5. Workflow management systems often have to deal with deviations and changes that were not anticipated at design-time. For example, people may stop performing certain roles or may take roles that normally they do not take. Similarly, deviations with respect to what a process model dictates may occur as a result for example of deadline violations or emergencies (for example a task may be skipped or fast-tracked because of a special situation). This kind of run-time flexibility is supported by systems such as YAWL (cf. chapter X) and ADEPT (cf. chapter X). Can you explain the difference between this so-called *run time flexibility* and the notion of configurable process model introduced in this chapter? Hint: consider the lifecycles shown at the top of Figure 2.

6. In this chapter, we focused on the representation of variability of control-flow elements in a process model, such as optional tasks and configurable control-flow connectors. Can you provide examples where the variability of a process model affects data-flow elements (e.g. a task that may or may not produce certain data objects as outputs), or resources (e.g. a task that may be performed by one role or by another depending on the variant being considered).

SUGGESTED ADDITIONAL READINGS

Configurable Nodes

1. Michael Rosemann and Wil M. P. van der Aalst: A configurable reference modelling language. Information Systems 32(1): 1-23, 2007. *This is the main article on C-EPCs. It describes the C-EPC notation and provides a technique to individualize C-EPCs.*

2. M. La Rosa, M. Dumas, A. ter Hofstede, J. Mendling, and F. Gottschalk. Beyond Control-Flow: Extending Business Process Configuration to Resources and Objects. QUT ePrints Technical Report #11240, 2007. *In this article configuration of organizational roles and objects participating in a process model is discussed. The process modelling notation used is C-iEPC - an extension of C-EPC, in which configurable nodes are extended to cater for variability of objects and roles.*

Hiding & Blocking

3. F. Gottschalk, W. van der Aalst, M. Jansen-Vullers, M. La Rosa. Configurable Workflow Models. International Journal on Cooperative Information Systems, 2008 (forthcoming). *In this article, the approach based on the hiding & blocking operators is applied to executable process languages, such as C-YAWL, BPEL and SAP Web Flow.*

4. W.M.P. van der Aalst, M. Dumas, F. Gottschalk, A.H.M. ter Hofstede, M. La Rosa and J. Mendling. Correctness-Preserving Configuration of Business Process Models. In Proceedings of Fundamental Approaches to Software Engineering (FASE 2008), Lecture Notes in Computer Science 4961, pages 46–61, Budapest, Hungary, 2008. Springer-Verlag. *In this article focus is on the issue of correctness of configured process models and it is shown under which circumstances a process model that results from a configuration can be guaranteed to be correct, from a syntactic and semantic perspective. The process models are represented as Petri nets and configuration is carried out by using the hiding & blocking operators.*

Annotation-based Process Variability

5. Process Family Engineering in Service-Oriented Applications (PESOA) at www.pesoa.de. *This web-site contains links to research in the area of annotation-based process variability. An Eclipse plugin can be downloaded from this tool, to configure feature diagrams and link their result to variant-reach process models.*

6. F. Puhlmann, A. Schnieders, J Weiland and M. Weske Variability Mechanisms for Process Models. Process Family Engineering in Service-Oriented Applications (PESOA). BMBF-Project. Technical report. 2005. *This is the technical report on the outcomes of the PESOA project.*

Questionnaire Models

7. Process Configuration home page at www.processconfiguration.com. *This web-site contains links to research in the area of process configuration. The Quaestio tool is also available for download from this web-site.*

8. M. La Rosa, W. van der Aalst, M. Dumas, and A. ter Hofstede. Questionnaire-based Variability Modeling for System Configuration. International Journal of Software and Systems Modeling, 2008 (forthcoming). *This article introduces the use of questionnaires for the configuration of*

process models. The article also compares the approach to related areas in the field of software engineering.

9. M. La Rosa, J. Lux, S. Seidel, M. Dumas and A. ter Hofstede. Questionnaire-driven Configuration of Reference Process Models. In Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007), Lecture Notes in Computer Science 4495, pages 424-438, Trondheim, Norway, 2007. Springer-Verlag. *This article provides a discussion of the application of the questionnaire-based approach to the configuration of C-EPCs.*

10. Krzysztof Czarnecki and Michal Antkiewicz: Mapping Features to Models: A Template Approach Based on Superimposed Variants. In Proceedings of the 4th International Conference on Generative Programming and Component Engineering, Lecture Notes in Computer Science 3676, pages 422-437, Tallinn, Estonia, 2005. Springer. *This paper shows another approach to link feature diagrams to process models represented as UML ADs.*

REFERENCES

VAN DER AALST, W. M. P. & BASTEN, T. (2002) Inheritance of Workflows: An Approach to Tackling Problems Related to Change. *Theoretical Computer Science,* 270**,** 125-203.

VAN DER AALST, W. M. P., DREILING, A., GOTTSCHALK, F., ROSEMANN, M. & JANSEN-VULLERS, M. H. (2006) Configurable Process Models as a Basis for Reference Modeling. *BPM 2005 Workshops (Workshop on Business Process Reference Models),* 3812**,** 512-518.

BATORY, D. S. (2005) Feature Models, Grammars, and Propositional Formulas. *Proceedings of the 6th International Conference on Software Product Lines (SPLC),* 3714**,** 7-20.

BECKER, J., DELFMANN, P., DREILING, A., KNACKSTEDT, R. & KUROPKA, D. (2004) Configurative Process Modeling - Outlining an Approach to increased Business Process Model Usability. *Proceedings of the 15th IRMA International Conference*.

BECKER, J., DELFMANN, P. & KNACKSTEDT, R. (2006) Adaptive Reference Modeling: Integrating Configurative and Generic Adaptation Techniques for Information Models. *Reference Modeling Conference 2006*.

BUNKE, H. (2000) Recent Developments in Graph Matching. *International Conference on Pattern Recognition (ICPR).* Barcelona, Spain, IEEE Computer Society.

CURRAN, T. & KELLER, G. (1997) *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*, Upper Saddle River.

GOTTSCHALK, F., AALST, VAN DER W. M. P. & JANSEN-VULLERS, M. H. (2007) Configurable Process Models - A Foundational Approach. *Reference Modeling. Efficient Information Systems Design Through Reuse of Information Models***,** 59-78.

GOTTSCHALK, F., VAN DER AALST, W. M. P., JANSEN-VULLERS, M. H. & LA ROSA, M. (2008) Configurable Workflow Models. *International Journal of Cooperative Information Systems,* (forthcoming).

JANSEN-VULLERS, M. H., VAN DER AALST, W. M. P. & ROSEMANN, M. (2006) Mining Configurable Enterprise Information Systems. *Data and Knowledge Engineering,* 56**,** 195-244.

LA ROSA, M., VAN DER AALST, W. M. P., DUMAS, M. & TER HOFSTEDE, A. H. M. (2008) Questionnaire-based Variability Modeling for System Configuration. *International Journal on Software and Systems Modeling,* (forthcoming).

LA ROSA, M., LUX, J., SEIDEL, S., DUMAS, M. & TER HOFSTEDE, A. H. M. (2007) Questionnaire-driven Configuration of Reference Process Models. *19th International Conference on Advanced Information Systems Engineering (CAiSE).* Trondheim, Norway, Springer-Verlag.

PUHLMANN, F., SCHNIEDERS, A., WEILAND, J. & WESKE, M. (2005) Variability Mechanisms for Process Models. Process Family Engineering in Service-Oriented Applications (PESOA). BMBF-Project.

ROSEMANN, M. & AALST, W. M. P. V. D. (2007) A Configurable Reference Modelling Language. *Information Systems,* 32**,** 1-23.

SCHNIEDERS, A. & PUHLMANN, F. (2006) Variability Mechanisms in E-Business Process Families. *Proceedings of the 9th International Conference on Business Information Systems (BIS'06)***,** 583-601.

SCHOBBENS, P.-Y., HEYMANS, P., TRIGAUX, J.-C. & BONTEMPS, Y. (2006) Feature Diagrams: A Survey and A Formal Semantics. *14th International Conference on Requirements Engineering.* Minneapolis, Minnesota, USA.

STEPHENS, S. (2001) The Supply Chain Council and the SCOR Reference Model. *Supply Chain Management - An International Journal,* 1**,** 9-13.